BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Pada bagian ini akan dilakukan analisis sistem yang bertujuan untuk mempelajari dan menganalisa kebutuhan sistem yang akan dibuat sehingga dapat dilakukan perancangan sistem dengan kriteria dan perangkat-perangkat yang ditentukan. Analisis dan perancangan berfungsi untuk mempermudah, memahami dan menyusun perancangan pada bab selanjutnya. Dari proses analisis akan dapat dihasilkan berbagai macam saran perbaikan terhadap sistem yang dapat dijadikan dasar dalam perancangan *Information Retrieval Sistem*.

3.1.1 Analisis Masalah

Skripsi merupakan penulisan karya ilmiah berisi hasil penelitian menyeluruh yang disusun secara sistematis berdasarkan ketentuan metode penelitian ilmiah. Secara garis besar abstrak jurnal skripsi merupakan atribut dari jurnal skripsi yang memberikan gambaran keseluruhan mengenai penelitian yang telah dilakukan. Untuk pencarian jurnal skripsi yang digunakan di perpustakaan khususnya di Universitas Muhammadiyah Gresik masih manual, meskipun di perpustakaan sudah menggunakan sistem perpustakaan (*catalog* dan *website*) yang memberikan kemudahan bagi pengguna dengan menginputkan kata kunci (*keyword*) berdasarkan subyek dan judul skripsi yang diinginkan. Pencarian tersebut memiliki kelemahan, yaitu dokumen yang dihasilkan dari pencari terkadang tidak sesuai dengan yang di inginkan pengguna. Oleh karena itu, permasalahan yang akan diteliti adalah membuat sistem pencarian abstraksi jurnal skripsi yang mirip dengan *query*.

3.1.2 Analisis Kebutuhan Fungsional Sistem

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem. Ada beberapa kebutuhan fungsional sistem yang akan dikembangkan untuk aplikasi pencarian abstraksi jurnal skripsi yaitu:

1. Sistem dapat melakukan pencarian dokumen abstrak jurnal skripsi yang relevan.

2. Sistem dapat mengelola dokumen abstrak jurnal skripsi menjadi *inverted index* dengan membentuk *term* pada setiap kemunculan dokumen tertentu.
3. Memberikan Peringkatan pada dokumen yang mirip dengan *query*.

3.1.3 Kebutuhan Pembuatan Sistem

A. Kebutuhan perangkat Lunak

Adapun perangkat lunak yang dibutuhkan dalam pembangunan aplikasi tersebut adalah sebagai berikut :

1. Microsoft Win XP SP3
2. Adobe Dreamweaver CS5
3. Mozilla Firefox
4. Editplus 3
5. SQLyog Enterprise
6. Xampp

B. Kebutuhan perangkat keras

Perangkat keras yang dibutuhkan memiliki spesifikasi sebagai berikut :

1. Prosesor Dual Core
2. RAM 1GB
3. HDD space 320GB
4. Monitor
5. Keyboard dan mouse

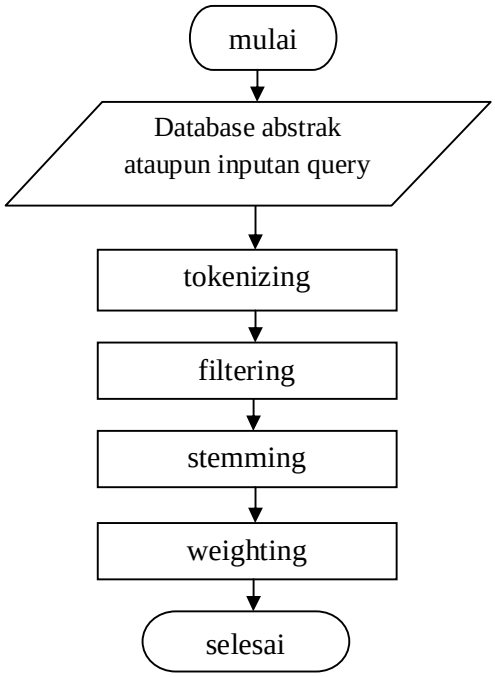
3.1.4 Deskripsi Sistem

Sistem yang dibangun adalah aplikasi atau tool pencarian abstrak jurnal skripsi menggunakan model ruang vektor. Tujuan dari sistem ini mengukur kemiripan dan relevansi untuk menemukan jurnal skripsi yang sesuai denga *query.* Secara umum sistem ini melalui tahapan Pra-pemrosesan/*indexing*, yaitu pada tahapan Pra-pemrosesan mencakup *case foding*, *tokenizing*, *filtering*, *stemming*, dan *weigthing*. Kemudian dilanjutkan dengan menyimpan setiap *term* yang penting ke dalam suatu index. Sistem ini akan menerima *query* dari pengguna kemudian memproses *query* tersebut dan sistem akan melakukan perhitungan

kemiripan antara *query* dengan daftar abstak yang tersedia, untuk menghasilkan kumpulan dokumen yang relevan dan terurut berdasarkan ranking kerelevanannya dengan *query* dari pengguna.

Di dalam sistem dilakukan *preprosesing* terhadap dokumen abstrak diantaranya *tokenizing*,*filtering, stemming* dan *weighting*.

Berikut diagram alir proses preprosesing dalam proses pengolahan isi abstrak dan inputan *query*.



Gambar 3.1 Diagram alir proses *preprosesing*

Proses yang terjadi pada gambar 3.1 dalam tahapan perprosesing yaitu:

1. Pengolahan terhadap dokumen abstrak dan inputan query didatabase.
2. Proses *tokenizing* yaitu memecah kalimat menjadi *terms*, Tahapan ini juga menghilangkan karakter-karakter tertentu seperti tanda baca dan mengubah semua token ke bentuk huruf kecil (*lower case*).
3. *Filtering* adalah pemilihan kata dan pencocokan kata terhadap stopword kemudian menghilangkan kata yang ada dalam stopword.
4. *Stemming* proses penghilangan awalan dan akhiran atau bisa disebut

juga kata-kata yang muncul di dalam dokumen sering mempunyai banyak varian morfologik.

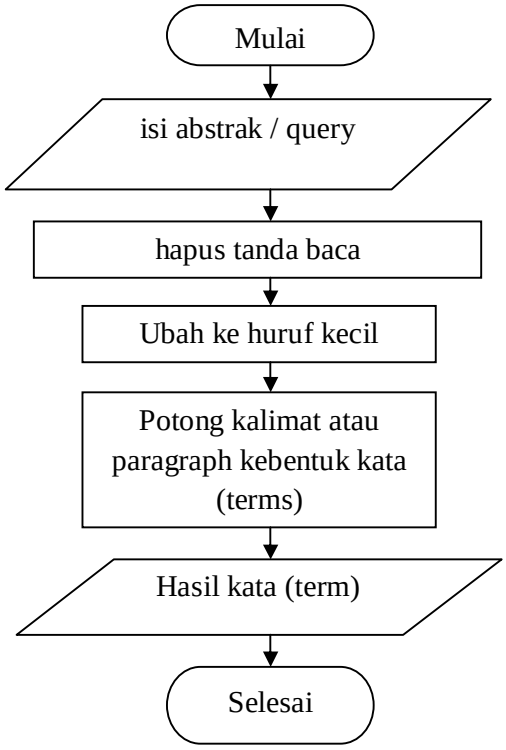
5. Kemudian melakukan penghitungan pembobotan tiap *terms*.

Dalam proses *preprosesing* dilakukan beberapa tahapan untuk melakukan pengindeksan dokumen abstrak diantaranya *tokenizing*,*filtration*, *stemming* dan *weighting*.

- **Tokenizing**

  Tokenizing adalah tugas memisahkan deretan kata di dalam kalimat, paragraf atau halaman menjadi token atau potongan kata. Tahapan ini juga menghilangkan karakter-karakter tertentu seperti tanda baca dan mengubah semua token ke bentuk huruf kecil.

  Berikut diagram alir *tokenizing* :



Gambar 3.2 Diagram alir *tokenizing*

Proses yang terjadi pada gambar 3.2 dalam tahapan *tokenizing* yaitu:

1. Sistem mencari abstrak dalam database.

2. Sistem menghilangkan tanda baca.
3. Sistem mengubah semua huruf menjadi huruf kecil.
4. Sistem memisah kalimat atau paragraph kebentuk kata (*term*).
5. Sistem mengambil hasil dari proses berupa kata (*term*) yang akan dip roses *filtering*.

   Contoh *tokenizing*:

   - Sebelum *tokenizing*;

Banyak aplikasi yang memanfaatkan pengolahan citra, diantaranya absensi dengan pengenalan wajah, absensi sidik jari, termasuk pengenalan citra uang. Mengingat gaya hidup masyarakat yang semakin aktif dan dinamis, sehingga membutuhkan otomatisasi dalam keseharian.

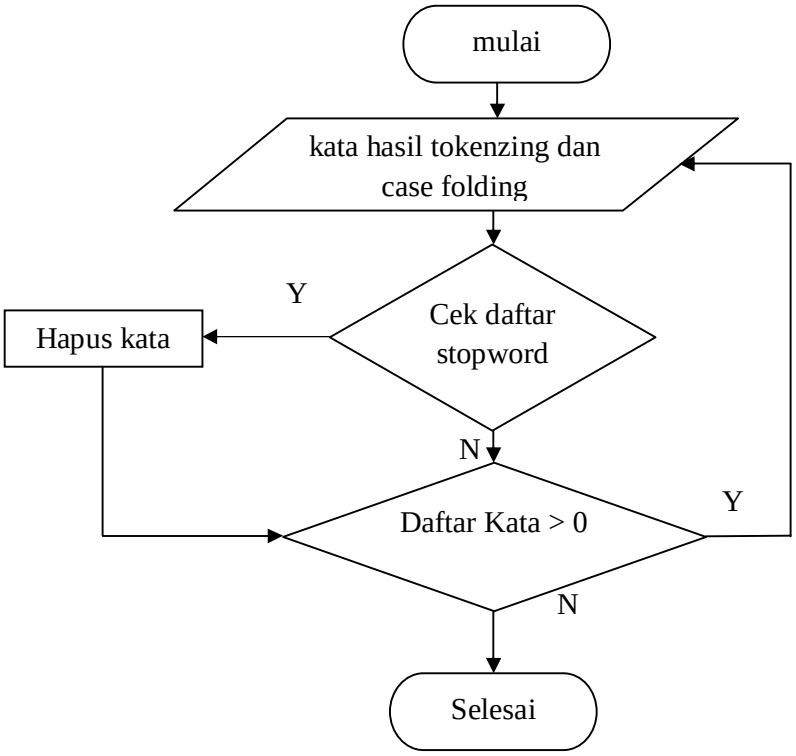
   - Sesudah *tokenizing*;

banyak	aplikasi	yang	memanfaatkan	pengolahan
citra	diantaranya	absensi	dengan	pengenalan
wajah	absensi	sidik	jari	termasuk
pengenalan	citra	uang	mengingat	gaya
hidup	masyarakat	yang	semakin	aktif
dan	dinamis	sehinggah	membutuhkan	otomatisasi
dalam	keseharian			

- **Filtering**

  Filtering adalah pemilihan kata dan pencocokan kata terhadap stopword kemudian menghilangkan kata yang ada dalam stopword.

  Berikut diagram alir dari proses *filtering* :



Gambar 3.3 Diagram alir dari proses *filtering*

Proses yang terjadi pada gambar 3.3 dalam tahapan *filtering* yaitu :

1. Sistem mengambil kata dari hasil *tokenizing dan case folding.*
2. Sistem melakukan pengecekan kata yang ada dalam stopword dalam database apakah ada kata yang sama atau tidak, jika ada yang sama maka sistem akan melakukan penghapusan, jika tidak ada maka kata tersebut akan disimpan untuk proses *stemming.*
3. Sistem akan melakukan pengecekan berulang-ulang sampai tidak ada kata yang sama.

   Contoh *filtering*:

- Seblum *filtering*;

- Setelah *filtering*;

3.2 Stemming

Algoritma Stemming Nazief Andriani

Stemming adalah pengubahan kata ke bentuk kata dasar atau penghapusan imbuhan. Dalam sistem ini digunakan algoritma *stemming* Nazief Andriani yang merupakan pembentukan kata dasar yang sudah dibahas didalam BAB II.

Pada umumnya kata dasar pada bahasa Indonesia terdiri dari kombinasi:

[[[AW+]AW+]AW+]kata dasar[[+AK][+KK][+P]]

AW : Awalan

AK : Akhiran

KK : Kata ganti kepunyaan

P : Partikel

Tanda kurung menandakan imbuhan bersifat opsional. Berikut ini contoh penerapan algoritma nazief dan andriani

Berkeliar**an**>

Berkeliar + **an (**AK**) >**hapus akhiran –**an**

Ber (AW) + **ke**liar > hapus awalan **–ber**

Ke (AW) + liar > hapus awalan –**ke**

Liar = kata dasar

Mempersembah**kannyapun**>

Mempersembah**kannya** + **pun** (P) > hapus partikel –**pun**

Memeprsembah**kan** + **nya** (KK) > hapus kata ganti kepunyaan **–nya**

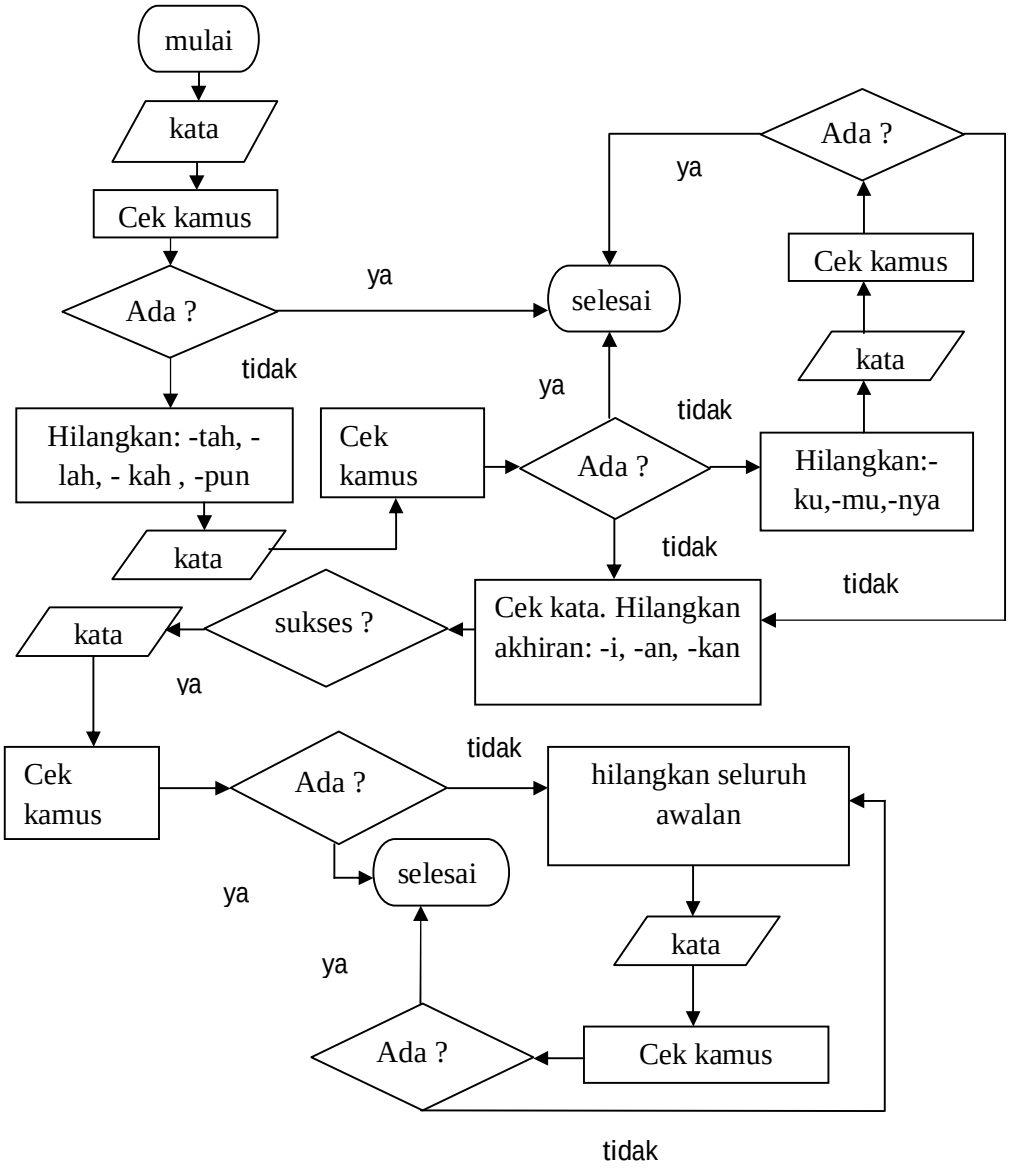
Mempersembah + **kan (**AK) > hapus akhiran **–kan**

Mem (AW) + **per**sembah > hapus awalan **-mem**

Per (AW) + sembah > hapus awalan **–per**

Sembah = kata dasa

Berdasarkan algoritma *stemming* nazief-andriani pada BAB II, berikut pada gambar 3.5 penjelasan dengan diagram alir algoritma stmming Nazief Andriani.



Gambar 3.4 Diagram Alir Algoritma *Stemming* bahasa indonesia Nazief-Andriani.

Proses tahapan *stemming* nazief dan andriani yang terjadi pada gambar 3.4:

1. Kata dicari di dalam daftar kamus. Bila kata tersebut ditemukan di dalam kamus maka dapat diasumsikan kata tersebut adalah kata dasar sehingga algoritma dihentikan.

2. Bila kata di dalam langkah pertama tidak ditemukan di dalam kamus, maka diperiksa apakah sufiks tersebut yaitu sebuah partikel (“-tah”,”-lah”,”-pun” atau “-kah”). Bila ditemukan maka partikel tersebut dihilangkan.
3. Pemeriksaan dilanjutkan pada kata ganti milik(“-ku”,”-mu”,”-nya”).bila ditemukan maka kata ganti tersebut dihilangkan.
4. Memriksa akhiran (“-i”,”-an”,”-kan”). Bila ditemukan maka akhiran tersebut dihilangkan.

   Hingga langkah ke-4 dibutuhkan ketelitian untuk memeriksa apakah akhiran “-an” merupakan hanya bagian dari akhiran “-kan” dan memeriksa lagi apakah partikel (“-tah”,”-lah”,”-pun” atau “-kah”) dan kata ganti milik (“-ku”,”-mu”,”-nya”) yang telah dihilangkan pada langkah 2 dan 3 bukan merupakan bagian dari kata dasar.
5. Memeriksa awalan (“se-“,”ke-“,”di-“,”te-“,”be-“,”pe-“,”me-“). Bila ditemukan, maka awalan tersebut dihilangkan. Pemeriksaan dilakukan dengan berulang mengingat adanya kemungkinan multi-prefik.Langkah ke-5 ini juga membutuhkan ketelitian untuk memeriksa kemungkinan peluluhan awalan, perubahan prefix yang disesuaikan dengan huruf awal kata dan aturan kombinasi prefix-sufix yang diperbolehkan.
6. Setelah menyelesaikan semua langkah diatas dengan sukses, maka algoritma akan mengembalikan kata dasar yang ditemukan.

Contoh *stemming*:

- *Stemming*;

aplikasi	memanfaatkan	pengolahan	citra	absensi
pengenalan	wajah	absensi	sidik	jari	termasuk
pengenalan	citra	uang	gaya	hidup	masyarakat
aktif	dinamis	membutuhkan	otomatisasi	keseharian

Term yang berubah oleh proses stemming:

memanfaatkan ⟹ manfaat

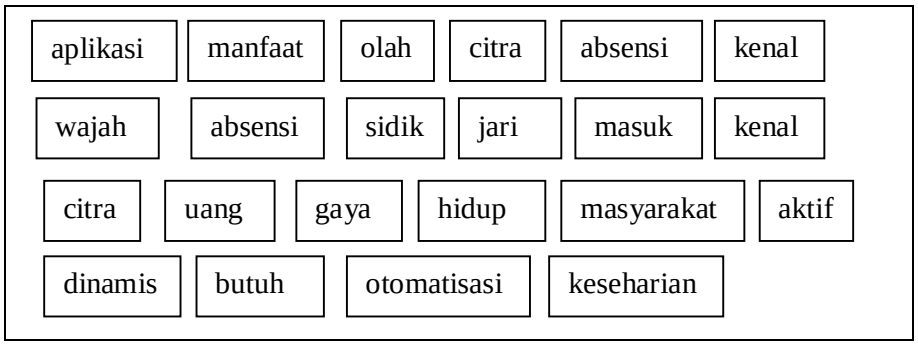
pengolahan ⟹ olah

termasuk ⟹ masuk

pengenalan ⟹ kenal

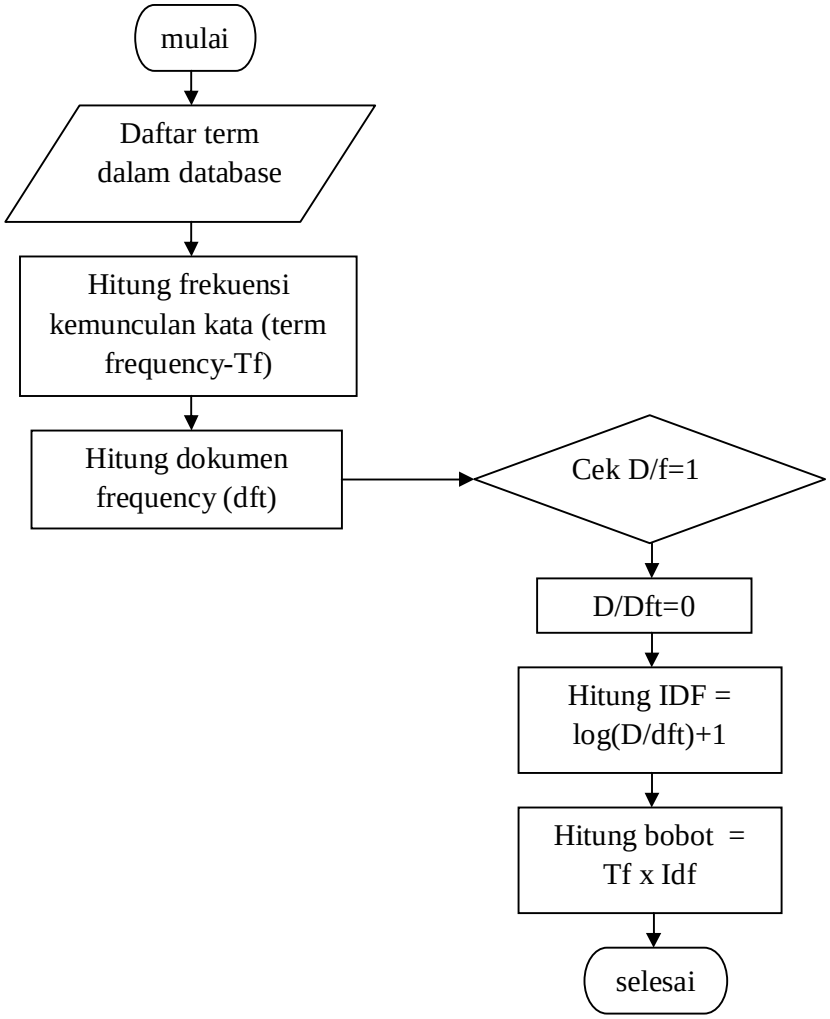
membutuhkan ⟹ butuh

- Setelah di *Stemming*;



3.3 Weighting (Pembobotan) TF-IDF

Metode TF-IDF merupakan metode pembobotan dalam bentuk sebuah metode yang merupakan integrasi antar *term frequency* (tf), dan inverse document frequency (idf) yang sudah dibahas pada BAB II. Dibawah ini penjelasan dengan diagram alir algoritma TF_IDF.



Gambar 3.5 Diagram Alir Pembobotan TF-IDF

Proses algoritma TF-IDF yang terjadi pada gambar 3.5.

1. Mengambil koleksi kata hasil proses *stemming* dari database
2. Menghitung jumlah kemunculan suatu kata (*term frequency*) perdokumen berdasarkan id dokumen
3. Hitung jumlah dokumen yang mengandung suatu kata tertentu
4. Mengecek nilai jumlah kemunculan dokumen yang mengandung suatu kata tertentu.
5. Membagi nilai dari total jumlah dokumen dengan nilai kemunculan dokumen bedasarkan kata tertentu

6. Menghitung nilai idf dengan cara menjumlah banyak dokumen dibagi dengan jumlah dokumen yang mengandung suatu kata. Kemudian di LOG dan dinormalisasi dengan menambah nilai 1.
7. Menghitung nilai bobot dengan mengalikan *term frequency* (Tf) dengan inverse document frequency (idf).

Contoh kasus terdapat tiga dokmen abstrak:

D1 : aplikasi manfaat olah citra absensi kenal wajah masuk citra uang

D2 : absensi sidik jari masuk kenal citra uang

D3 : gaya hidup masyarakat aktif dinamis butuh otomatisasi keseharian

Q : olah citra

Ket :

tf :*term frequency*

D : 3 (Jumlah Dokumen)

Q : query

Dft : Banyaknya dokumen yang mengandung term

N/dft : Dimana N jumlah dokumen yang diproses dibagi dengan kemunculan term

IDF : log(N/dft+1)

Tabel 3.1 Perhitungan TF-IDF

term	**tf**				**dft**	**N/dft**	**IDF**
	D1	**D2**	**D3**	**Q**			
aplikasi	1	0	0	0	1	3	1.48
manfaat	1	0	0	0	1	3	1.48
olah	1	0	0	1	1	3	1.48

Lanjutan **tabel 3.1** perhitungan TF-IDF

term	tf				dft	N/dft	IDF
	D1	D2	D3	Q			
citra	2	1	0	1	2	1.5	1.18
absensi	1	1	0	0	2	1.5	1.18
kenal	1	1	0	0	2	1.5	1.18
wajah	1	0	0	0	1	3	1.48
sidik	0	1	0	0	1	3	1.48
jari	0	1	0	0	1	3	1.48
masuk	1	1	0	0	2	1.5	1.18
uang	1	1	0	0	2	1.5	1.18
gaya	0	0	1	0	1	3	1.48
hidup	0	0	1	0	1	3	1.48
masyarakat	0	0	1	0	1	3	1.48

Tabel 3.2 Pembobotan TF-IDF

Term	W			
	D1	D2	D3	Q
Aplikasi	1.48	0	0	0
Manfaat	1.48	0	0	0
Olah	1.48	0	0	1.48
Citra	2.36	1.18	0	1.18
Absensi	1.18	1.18	0	0
Kenal	1.18	1.18	0	0
Wajah	1.48	0	0	0
Sidik	0	1.48	0	0
Jari	0	1.48	0	0
Masuk	1.18	1.18	0	0

Lanjutan **table 3.2** Pembobotan TF-IDF

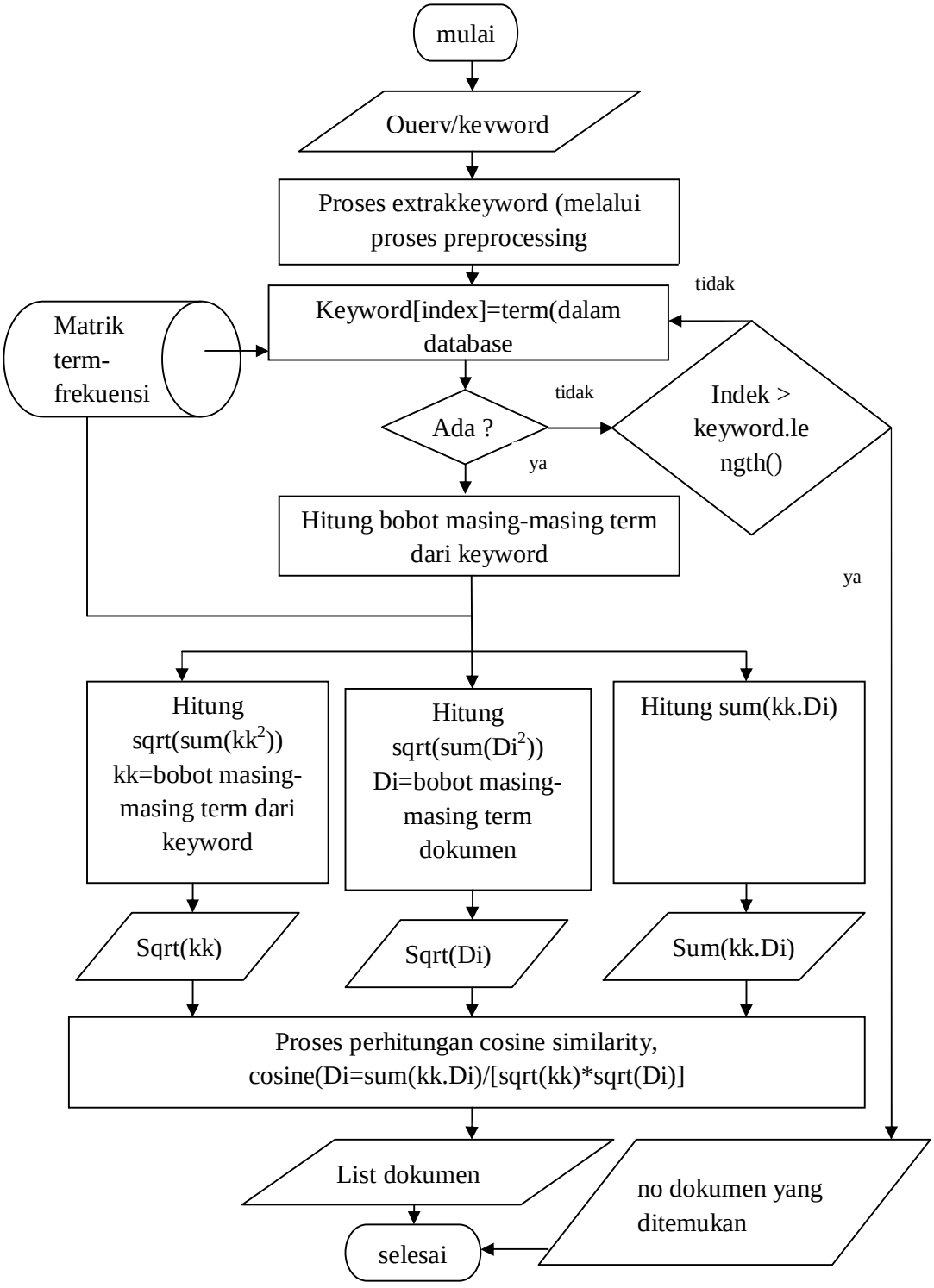
Term	W			
	D1	**D2**	**D3**	**Q**
Uang	1.18	1.18	0	0
Gaya	0	0	1.48	0
Hidup	0	0	1.48	0
masyarakat	0	0	1.48	0
Aktif	0	0	1.48	0
Dinamis	0	0	1.48	0

3.4 Algoritma *Vektor Space Model* (Model Ruang Vektor)

Vektor Space Model (Model ruang Vektor) adalah model sistem temu balik informasi yang mengukur kemiripan antara suatu dokumen dengan *query*.

Model ruang vektor menentukan kemiripan antara dokumen dengan *query* dengan cara merepresentasikan dokumen dan *query* masing-masing ke dalam bentuk vector,yang telah di jelaskan di BAB II,

berikut penjelasan pada gambar 3.6 Diagram Alir vector space model (model ruang vektor).



Gambar 3.6 Diagram Alir vector space model (model ruang vektor).

Proses algoritma *vector space model* yang terjadi pada gambar 3.6 :

1. Input *query / keyword* pada sistem
2. Sistem akan mengelolah kata yang diinputkan pengguna menjadi term
3. Sistem akan mengecek ke database, jika ada term maka akan dilakukan pembobotan dari masing-masing *keyword*. Jika tidak ada maka proses pencarian tidak dilakukan (dokumen tidak ditemukan)
4. Setelah dilakukan pembobotan pada tiap *term* kata kunci, selanjutnya dijalankan 3 proses dalam mendapatkan variable-varabel untuk menghitung tiap kemiripan dokumen.
5. Hasil jumlah (Summary) nilai kuadrat bobot dari keyword disimpan dalam variable kk, penjumlahan dari nilai kuadrat bobot masing-masing term dokumen disimpan di variabel Di, dan terakhir summary dari hasil perkalian nilai bobot dari keyword dan nilai bobot term terhadap dokumen disimpan pada variable sum(kk*Di). Nilai dari akar kata kunci "sqrt(kk)" dan akar dari Di "sqrt(Di)", serta nilai dari sum(kk*Di) menjadi variable-variabel dalam proses perhitungan cosine similarity untuk tiap dokumen terhadap keyword. Nilai dari cosine similarity untuk tiap dokumen dilakukan proses indexing secara descending yakni dari nilai yang terbesar hingga yang terkecil. Terakhir ditampilkan list dokumen abstrak.

Tabel 3.3 Perhitungan dot dokumen dan query

term	WD1 x Wq	WD2 x Wq	WD3 x Wq
aplikasi	0	0	0
manfaat	0	0	0
olah	2.1904	0	0
citra	2.7848	1.3924	0
absensi	0	0	0
kenal	0	0	0
wajah	0	0	0
sidik	0	0	0
jari	0	0	0

Lanjutan **table 3.3** Perhitungan dot dokumen dan query

term	WD1 x Wq	WD2 x Wq	WD3 x Wq
masuk	0	0	0
uang	0	0	0
gaya	0	0	0
hidup	0	0	0
masyarakat	0	0	0
aktif	0	0	0
dinamis	0	0	0
butuh	0	0	0
otomatisasi	0	0	0
keseharian	0	0	0
KK . D	4.9752	1.3924	0

Tabel 3.4 Perhitungan kuadrat bobot dokumen dan query

term	WD_1^2	WD_2^2	WD_3^2	Wq^2
aplikasi	2.1904	0	0	0
manfaat	2.1904	0	0	0
olah	2.1904	0	0	2.1904
citra	5.5696	1.3924	0	1.3924
absensi	1.3924	1.3924	0	0
kenal	1.3924	1.3924	0	0
wajah	2.1904	0	0	0
sidik	0	2.1904	0	0
jari	0	2.1904	0	0
masuk	1.3924	1.3924	0	0
uang	1.3924	1.3924	0	0
gaya	0	0	2.1904	0
hidup	0	0	2.1904	0
masyarakat	0	0	2.1904	0
aktif	0	0	2.1904	0

Lanjutan **table 3.4** Perhitungan bobot dokumen kuadrat dan query

term	WD_1^2	WD_2^2	WD_3^2	Wq^2
dinamis	0	0	2.1904	0
butuh	0	0	2.1904	0
otomatisasi	0	0	2.1904	0
keseharian	0	0	2.1904	0

Tabel 3.5 Perhitungan cosine similarity

$\sum\left(Wd_i^2, Wq^2\right)$	19.793074	11.3428	17.5232	2.5828
$\sqrt{\sum\left(Wd_i^2, Wq^2\right)}$	4.448940773	3.367907362	4.18607214	1.892828571
$\|d_i\| \times \|q\|$	7.246739379	6.37487128	7.92351696	0
$\frac{d_i \bullet q}{\|d_i\| \times \|q\|}$	0.589201703	0.218420096	0	0

Tabel 3.6 hasil perangkingan

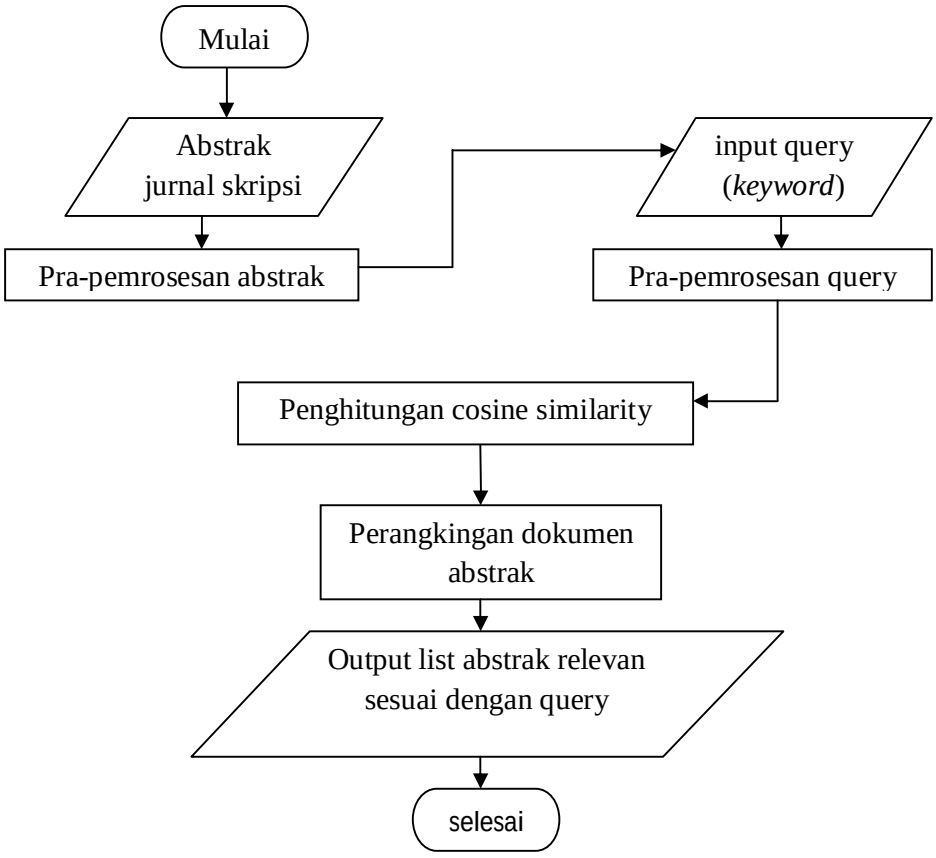
Rangking	Dokumen ke-	Similarity
1	D1	0.589201703
2	D2	0.218420096

Dari hasil pencarian dari tiga dokumen dengan query pengolahan citra yang ditemukan dua dukumen yang ditemukan dan yang tidak ditemukan hanya satu .

3.5 Perancangan Sistem

3.5.1 *flowchart* Sistem

Berikut ini adalah *flowchart* umum sistem pencarian jurnal skripsi yang di jelaskan pada Gambar 3.7:



Gambar 3.7 Flowchart system pencarian jurnal skripsi

Berikut penjelasan *flowchart* sistem pencarian jurnal skripsi Gambar 3.7:

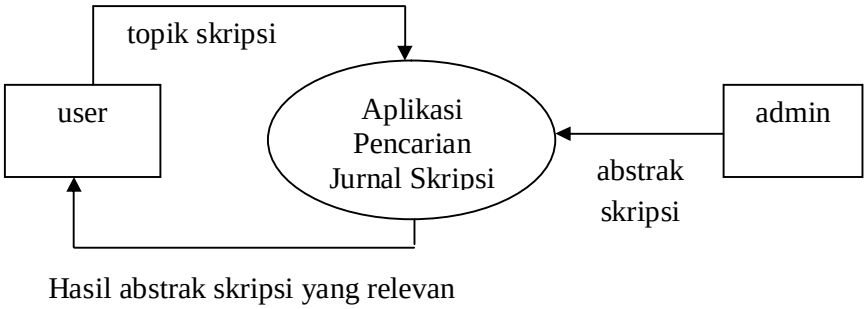
1. Sistem mengambil dan membaca daftar abstrak yang tersedia dalam database.
2. Melakukan Pra-pemrosesan terhadap abstrak, yang meliputi *tokenizing*, *filtering*, *stemming dan weigting (tf-idf)* sehingga didapatkan *terms* dalam bentuk akar yang nantinya dilakukan pengindeksan.
3. Sistem melakukan proses Pra-pemrosesan query yang meliputi

tokenizing, *filtering*, *stemming dan weigting (tf-idf)* hingga didapatkan *terms query*, dan Kemudian disipan dalam database.

4. Sistem malakukan perhitungan kemiripan (*similarity*) antara *terms query* dengan *term* dokumen abstrak.
5. Sistem melakukan perangkingan terhadap dokumen yang sesuai dengan *query* sesuai dengan nilai *similarity.*
6. Sistem mengeluarkan berupa list abstrak yang relevan sesuai dengan inputan *query*.

3.5.2 Diagram Konteks

Berdasarkan dari arus proses folwchart yang telah dijelaskan diatas, maka sistem dapat dijelaskan dengan diagram konteks sebagai berikut:



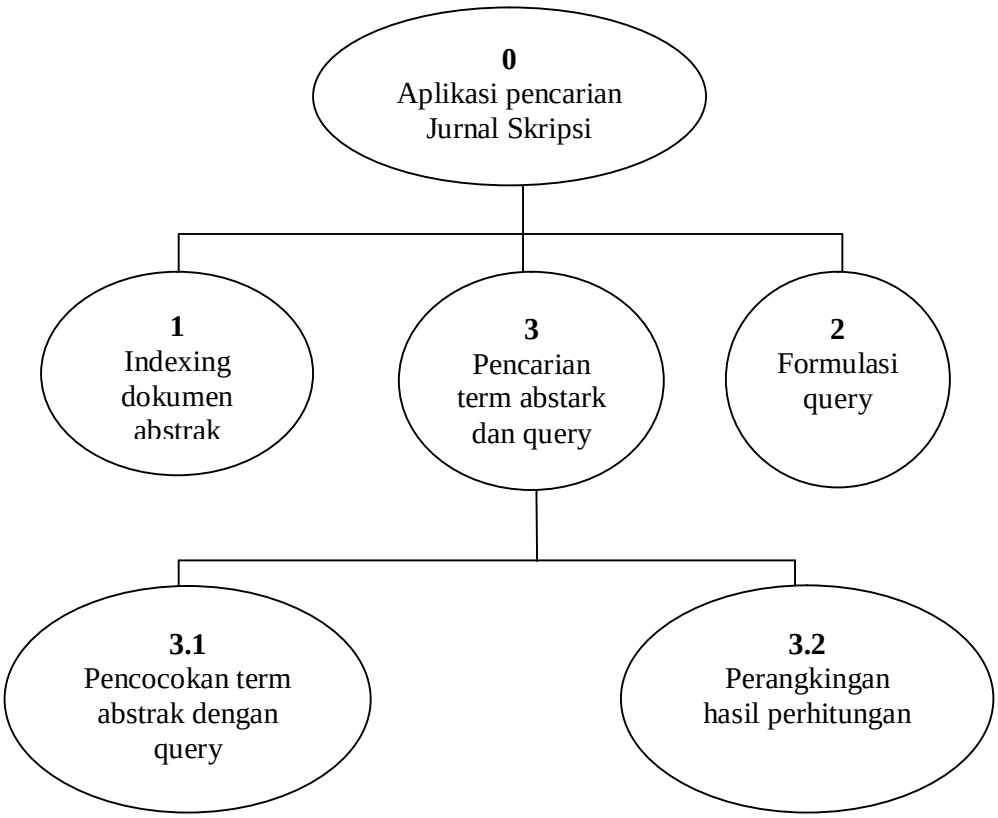
Gambar 3.8 Diagram Konteks Aplikasi Pencarian Jurnal Skripsi

Proses yang terjadi pada gambar 3.8 dalam tahapan diagram konteks aplikasi pencarian jurnal skripsi yaitu :

- Admin menginputkan abstrak skripsi ke database system pencarian jurnal skripsi.
- Pengguna menginputkan *query* berupa topik skripsi ke dalam sistem pencarian, kemudian mendapatkan hasil berupa abstrak skripsi yang relevan dan mirip dengan *query*

3.5.3 Diagram Berjenjang

Sebelum tahap selanjutnya, akan dilakukan pembuatan diagram berjenjang pada sistem pencarian jurnal skripsi yang berfungsi untuk menggambarkan proses jalannya pada sistem.



Gambar 3.9 Diagram Berjenjang Aplikasi Pencarian Jurnal Skripsi

Proses yang terjadi pada gambar 3.9 dalam tahapan diagram berjenjang aplikasi pencarian jurnal skripsi yaitu :

1. Top Level : Aplikasi pencarian jurnal skripsi.

1.1. Level 1 : merupakan turunan dari top level yang terdiri beberapa sub proses yaitu:

- Indexing dokumen abstrak.

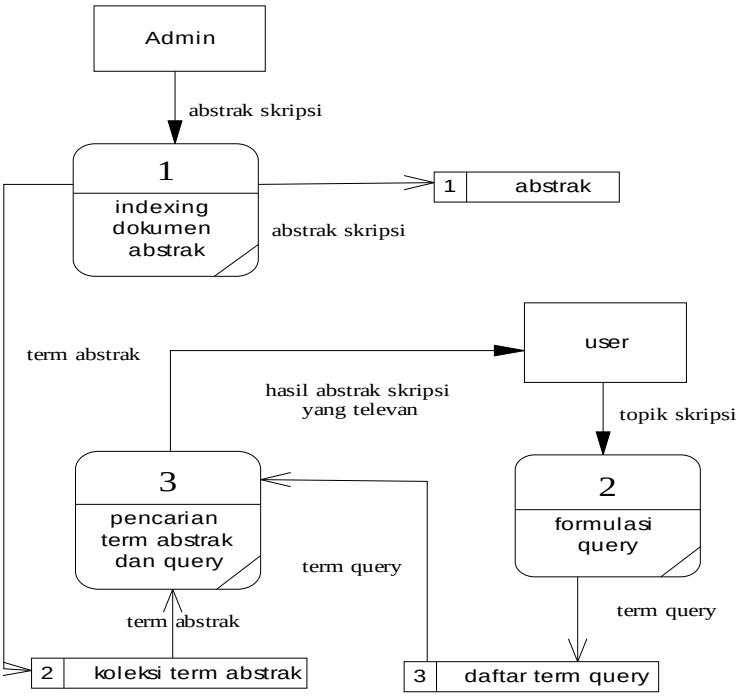
- formulasi *query*.
- Pencarian *term* abstrak dan *query*.

2. Level 2 : . Pencarian *term* abstrak dan *query*.
   - Pencocokan *term* abstrak dengan *query*.
   - Perhitungan vector dokumen dan *query*.
   - Perangkingan dokumen hasil dari perhitungan vector dokumen dan vector *query*.
   - Menampilkan hasil perangkingan.

3.5.4 Data Flow Diagram (DFD)

3.5.4.1 Data Flow Diagram (DFD) Level 1

Inputan pengguna berupa judul, nama penulis dan isi abtraksi skripsi tersebut.



Gambar 3.10 Data Flow Diagram Level 1

Proses yang terjadi pada gambar 3.10 dalam tahapan data *flowdiagram* level 1:

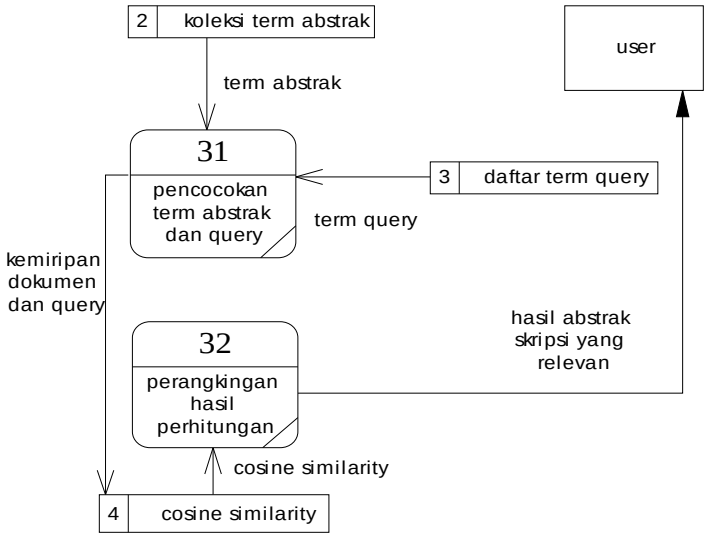
1. Sistem melakukan proses *indexing* abstrak didatabase yang sudah diinputkan admin kedatabase sistem menjadi *term* abstrak, kemudian

sistem menyimpan kedatabase dalam bentuk *term* yang sudah di *stemming* dan dihitung bobotnya menggunakan TF-IDF.

2. Sistem melakukan formulasi *query* yang telah diinputkan user berupa topik skripsi menjadi *term query*, kemudian sistem menyimpan kedatabase dalam bentuk *term* yang sudah di *stemming* dan dihitung bobotnya menggunakan TF-IDF.
3. Sistem melakukan pencarian *term* abstrak dan *term query*, setelah proses indexing dokumen dan formulasi query yang meliputi proses pencocokan term abstrak dengan query dengan model ruang vector,dilanjutkan perangkingan dengan cosine similarity Kemudian menghasilkan abstrak skripsi yang relevan mirip dengan *query*.

3.5.4.2 Data Flow Diagram (DFD) Level 2

Dari tahapan proses data flow diagram pencarian term asbtrak dan query dapat dijabarkan lebih detail dalam DFD level 2. Data Flow Diagram (DFD) Proses Pencarian abstrak dan query

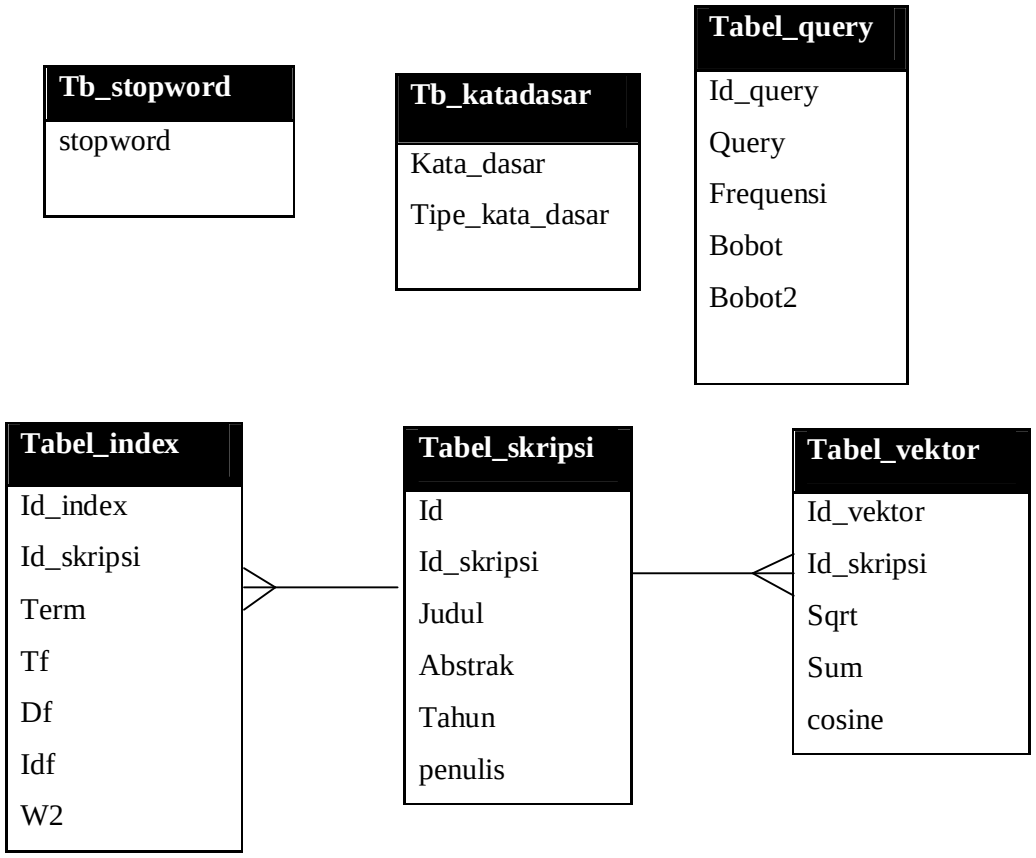


Gambar 3.11 Data Flow Diagram Level 2 pencarian term abstrak dan query

Proses yang terjadi pada gambar 3.11 dalam tahapan data flow diagram level 2 Tahap proses pencarian :

1. Sistem melakukan pencocokan kata (*term)* abstrak dan kata (*term) query*. Sistem melakukan perhitungan *vector* dokumen dan *vector query* terhadap *Term* dokumen dan *term query* dengan *cosine similarity*. Kemudian menghasilkan dokumen yang mirip *query*.
2. System melakukan perangkingan terhadap dokumen yang mirip *query*.
3. System menampilkan abstrak yang mirip query sesuai hasil perangkingan. Kemudian user menemukan hasil abstrak skripsi yang relevan.

3.6 Perancangan Database



Gambar 3.12 Entity Relational Diagram Aplikasi Pencarian Jurnal Skripsi

Berdasarka gambar 3.12 berikut penjelasan table-tabel yang digunakan dalam pembangunan apliksi pencarian jurnal skripsi :

1. Tabel stopword (stoplist)

   Tabel stopword berfungsi untuk menyimpan daftar stoplist seperti: “yang”,”di”,”dan” yang nantinya digunakan untuk proses *filtering* pada tahapan Pra-pemrosesan. Kata_stopword yang digunakan untuk penyimpanan kata yang akan dihilangkan pada proses *filtering*. Sehingga dapat dilakukan pengecekan terhadap tiap kata dari proses *tokenizing*, jika kata tersebut ditemukan dalam Tabel stopword ini maka kata tersebut dihilangkan dan tidak diproses ke proses selanjutnya. Berikut Struktur dari Tabel stopword adalah :

Tabel 3.7 Struktur tabel tb_stopwords

Nama field	Tipe	Keterangan
stopword	varchar (50)	

2. Tabel kata dasar

   Tabel kata dasar berisi kumpulan kata dasar berdasarkan kamus bahasa Indonesia untuk digunakan dalam proses *stemming* setelah proses *filtering* pada tahapan Pra-pemrosesan. Dalam Table ini juga terdapat field tipe_kata_dasar yang memberikan keterangan tipe dari kata dasar tersebut apakah berupa kata kerja ataukah kata sifat. Jika kata dari proses *filtering* tersebut tidak terdapat dalam daftar kata dasar maka akan dilakukan proses *stemming* terhadap kata tersebut. Adapun struktur dari Table katadasar adalah sebagai berikut :

Tabel 3.8 Struktur tabel tabel kata dasar

Nama_Field	Tipe	Keterangan
Kata_dasar	varchar (50)	
Tipe_kata_dasar	Varchar (20)	

3. Tabel *query*

   Tabel tabel_query digunakan untuk penyimpanan sementara *query* inputan pengguna yang telah dilakukan Pra-pemrosesan dan bobot dari tiap *term* dalam *query* untuk nantinya digunakan dalam proses perhitungan cosine *similarity* dengan *term* documents. Adapun struktur tabel_query adalah sebagai berikut:

Tabel 3.9 Struktur tabel tabel_query

Nama_Field	Tipe	Keterangan
Id_query	varchar (50)	
query	varchar (50)	Term query
frekuensi	In (11)	Jumlah tiap term
Bobot	Double	Bobot dari tiap term
Bobot2	Double	Nilai kuadrat bobot tiap term

4. Tabel index

   Tabel index berfungsi untuk penyimpanan daftar *term* abstrak hasil proses Pra-pemrosesan, nilai TF (*term* frequency) masing-masing *term* dari tiap document, nilai IDF (inverse document frequency) serta nilai bobot tiap *term* yang didapat dari perhitungan perkalian tf dan idf. Tabel ini memiliki struktur sebagai berikut :

Tabel 3.10 Struktur tabel tabel_index

Nama_Field	Tipe	Keterangan
Id_index	Int (11)	
Id_skripsi	varchar (100)	Term query
Term	Varchar (50)	Kata hasil Pra-pemrosesan
Tf	In (11)	Nilai frekuensi tiap term
Df	Int (11)	Nilai banyaknya dokumen tiap term
Idf	Double	Inverse dokumen frekuensi
W	Double	Bobot tiap term
W2	Double	Nilai kuadrat bobot tiap term

5. Tabel skripsi

   Tabel skripsi berfungsi untuk penyimpanan daftar abstrak skripsi, yaitu judul skripsi, abstrak, nama penulis dan tahun pembuatan. Judul skripsi disimpan pada field judul_skripsi, narasi abstrak disimpan pada field abstrak, nama penulis disimpan pada field penulis, sedangkan tahun pembuatan skripsi terdapat pada field tahun.:

Tabel 3.11 Struktur tabel tabel_skripsi

Nama_Field	Tipe	Keterangan
Id	Int (11)	
Id_skripsi	varchar (100)	Term query
judul	Varchar (200)	Judul skripsi
Abstrak	Tex	Abstrak skripsi
Tahun	Year (4)	Tahun pembuatan skripsi
Penulis	Varchar(50)	Penulis atau pembuat skripsi

6. Tabel Vektor

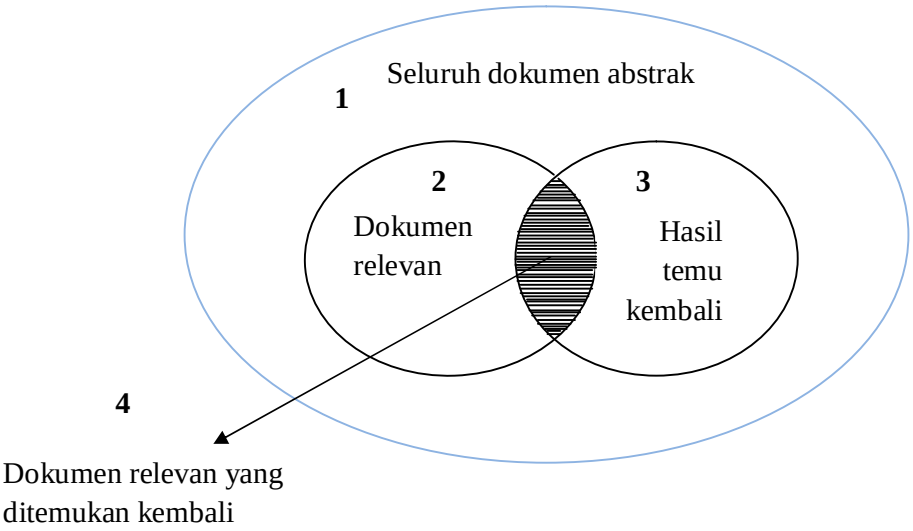
   Tabel tabel_hitung_cosine berfungsi dalam perhitungan *similarity* tiap dokumen terhadap *query*, untuk mempermudah dalam perhitungan cosine *similarity* dan perankingan dalam mendapatkan hasil dokumen yang memliki nilai *similarity* tertingggi.

Tabel 3.12 Struktur tabel tabel_vektor

Nama_Field	Tipe	Keterangan
Id_vektor	Int (11)	
Id_skripsi	varchar (100)	Id skripsi
sqrt	Double	Panjang vector dokumen
Sum	Double	Jumla dari perkalian bobot term query dengan term dokumen
Cosine	Double	Nilai persamaan query dengan dokumen menggunakan perhitungan cosine

3.7 Evaluasi Sistem Pencarian Jurnal Skripsi

Untuk mengukur evaluasi kinerja sistem temu kembali informasi, yaitu menggunakan Recall dan Precission. *Recall* adalah rasio antara dokumen yang ditemukembalikan relevan dari seluruh dokumen yang seharusnya ditemukembalikan relevan yang ada di dalam sistem, sedangkan *precision* adalah rasio dokumen relevan yang berhasil ditemukembalikan dari seluruh dokumen yang berhasil ditemu-kembalikan.



Gambar 3.13 *Recall* dan *Precision*

Berdasarkan gambar 3.13 :

1. Seluruh dokumen abstrak yang ada di database.
2. Dokumen yang tidak ditemukan tapi relevan.
3. Dokumen yang ditemukan tapi tidak relevan
4. Dokumen relevan yang berhasil ditemukan

Tabel 3.13 tabel ketergantungan

	Relevant	nonRelevant
Retrieved	True positive (tp)	False positive (fp)
Non retrieved	False negative (fn)	True negative (tn)

Ket :

Tp = Dokumen yang berhasil ditemukan dan relevant.
Fn = Dokumen yang tidak berhasil ditemukan dan relevant.
Tn = Dokumen yang tidak berhasil ditemukan dan tidak relevant.
Fp = Dokumen yang berhasil ditemukan dan tidak relevant.

Rumus menentukan precission :

$$\Pr ecission = \frac{tp}{(tp + fp)}$$

Persamaan (3.1)

Rumus menentukan nilai Recall :

$$\mathrm{Re}\, call = \frac{tp}{(tp + fn)}$$

Persamaan (3.2)

Rumus menentukan nilai Akurasi :

$$Accuracy = \frac{tp + tn}{tp + fp + tn + fn}$$

Persamaan (3.3)

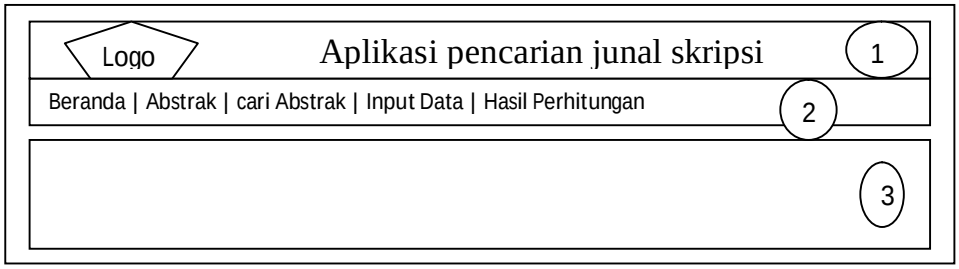
Nilai Precision, Recall dan Accuracy dinyatakan dalam persen. Semakin tinggi ketiga nilai tersebut menunjukkan semakin baiknya kinerja aplikasi. Evaluasi yang akan dilakukan dalam penelitian ini adalah menghitung nilai dari precision, recall dan accuracy berdasarkan dokumen yang ditemukan oleh aplikasi.Sedangkan untuk menentukan nilai dari *recall, precision* dan a*ccurancy* harus didapatkan jumlah dokumen yang relevan terhadap suatu topik dokumen abstrak.

Menurut Rijsbergen (1979) relevansi merupakan sesuatu yang bersifat subyektif. Setiap orang mempunyai perbedaan untuk mengartikan sesuatu dokumen yang relevan terhadap sebuah topik informasi.

3.8 Perancangan Interface

3.8.1 Antarmuka Halaman Awal

Halaman awal sistem sebagai antarmuka pencarian jurnal skripsi seperti ditunjukkan pada. gambar 3.14 dibawah ini.



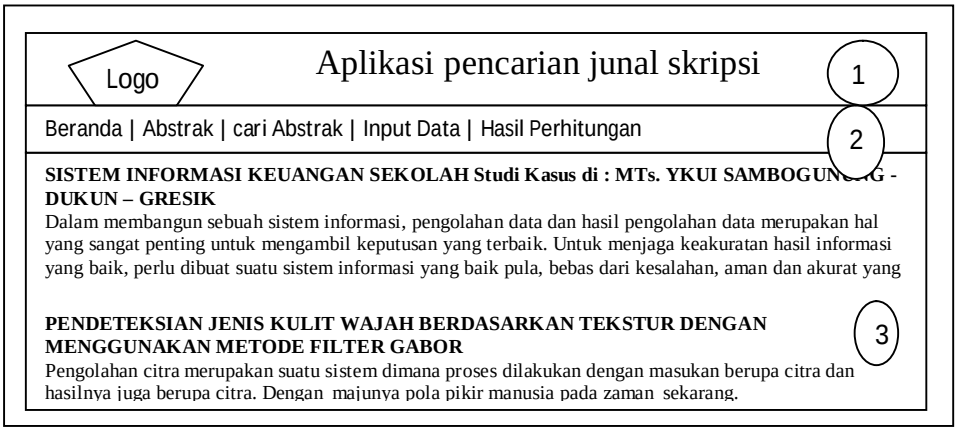
Gambar 3.14 Antarmuka halaman awal sistem

Berdasarkan gambar 3.14 :

1. Header dan logo civitas akademika dan judul aplikasi
2. Label menu, yang terdiri *beranda, tampilkan corpus* dan *pencarian jurnal skripsil.*
3. Deskripsi tentang aplikasi pencarian akan ditampilkan disini.

3.8.2 Antarmuka Halaman Tampilan Abstrak

Halaman ini merupakan tampilan bagi pengguna yang menampilkan seluruh abstrak skripsi program studi teknik informatika dari tahun 2011-2012. gambar 3.15 dibawah ini:



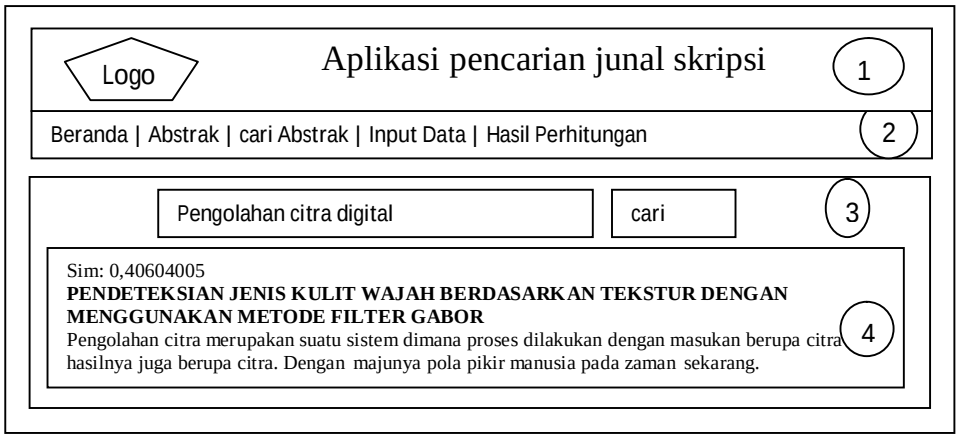
Gambar 3.15 Antarmuka tampilan abstrak

Berdasarkan gambar 3.15:

1. Logo civitas akademika dan judul aplikasi.
2. Label menu.
3. Abstrak skripsi yang ada dalam database.

3.8.3 Antarmuka Pencarian Skripsi

Tampilan pencarian bagi pengguna untuk mencari jurnal skripsi yang diinginkan dengan menginputkan *query*.



Gambar 3.16 Antarmuka Pencarian Skripsi

Berdasarkan gambar 3.16 :

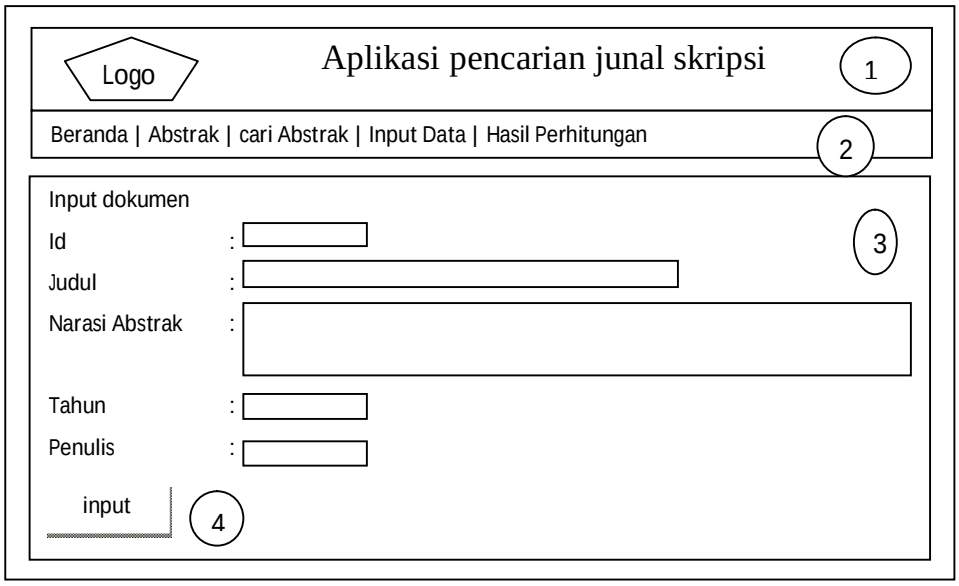
1. Logo civitas akademika dan judul aplikasi
2. Label menu
3. Label inputan *query* untuk mencari dokumen jurnal yang diinginkan.
4. Hasil pencarian skripsi

3.8.4 Antarmuka Input Data

Tampilan input data berguna untuk menginputkan data abstrak, data stopword dan kata dasar.

A. Input data abstrak

User dapat menginputkan data abstak ke dalam system pencarian



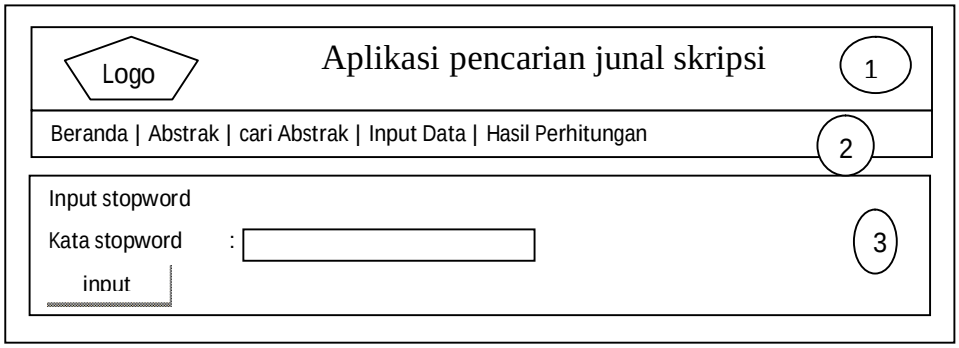
Gambar 3.17 Antarmuka input data abstrak

Berdasarkan gambar 3.17 :

1. Logo civitas akademika dan judul aplikasi
2. Label menu
3. Label inputan narasi abstrak
4. Tombol input data abstrak

B. Input data stopword

User dapat menginputkan data stopword guna untuk menambahkan kata-kata yang tidak penting.

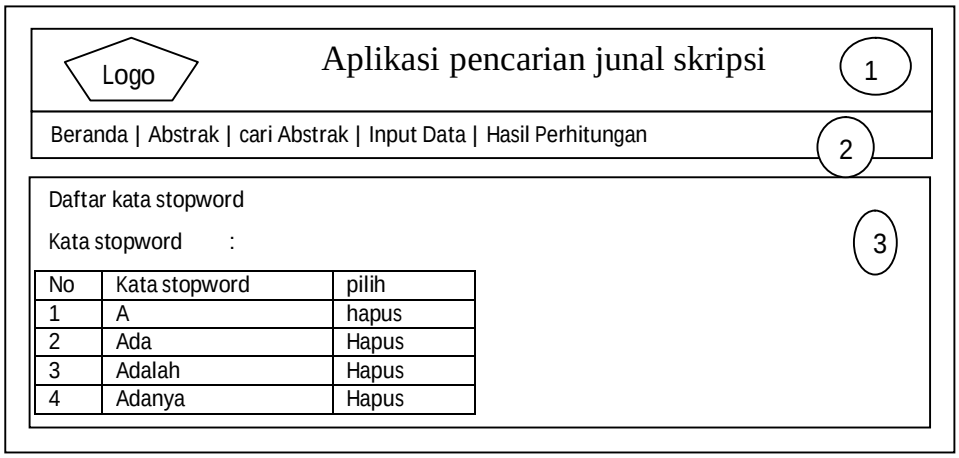


Gambar 3.18 Antarmuka input data stopword

Berdasarkan gambar 3.18 :

1. Logo civitas akademika dan judul aplikasi
2. Label menu
3. Label inputan kata stopword
4. Tombol input data stopword

- Daftar kata stopword



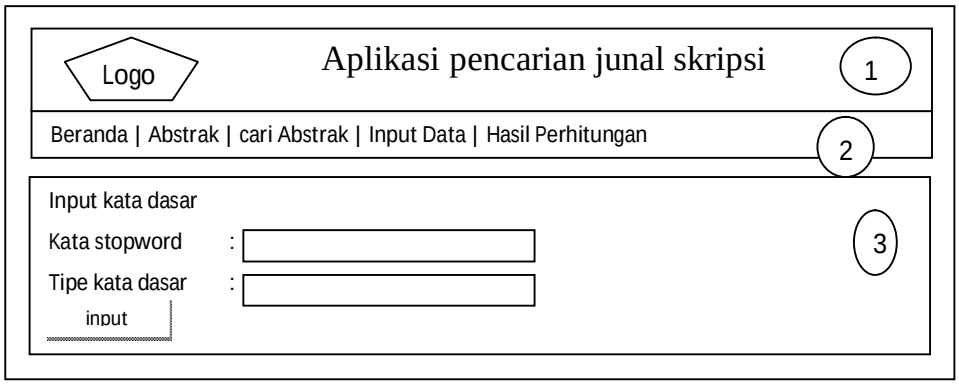
Gambar 3.19 Antarmuka daftar kata stopword

Berdasarkan gambar 3.19 :

1. Logo civitas akademika dan judul aplikasi
2. Label menu
3. Label daftar kata stopword

C. Input data kata dasar

User dapat menginputkan kata dasar guna untuk menambahkan kata dasar sesui ejaan bahasa indonesia.

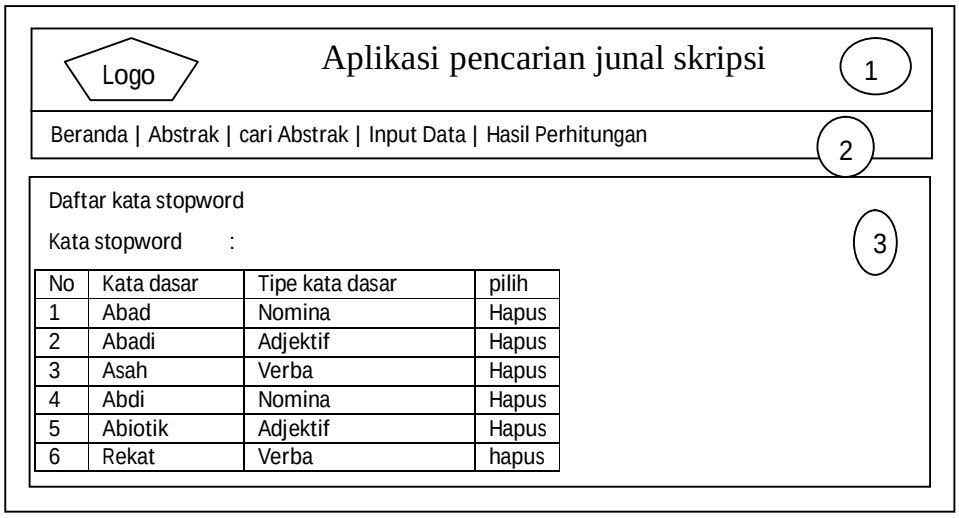


Gambar 3.20 Antarmuka input data kata dasar

Berdasarkan gambar 3.20 :

1. Logo civitas akademika dan judul aplikasi
2. Label menu
3. Label inputan kata dasar
4. Tombol input data kata dasar

- Daftar kata dasar



Gambar 3.21 Antarmuka daftar kata dasar

Berdasarkan gambar 3.21 :

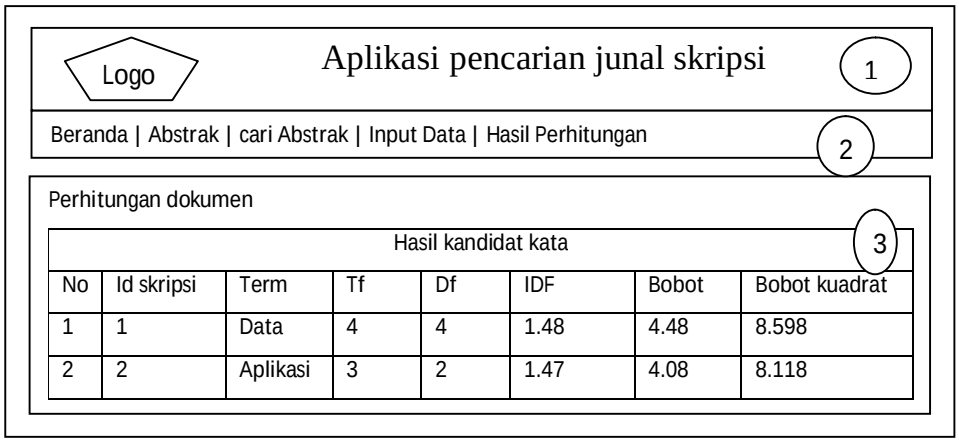
1. Logo civitas akademika dan judul aplikasi
2. Label menu
3. Label daftar kata dasar

3.8.5 Antarmuka Hasil Perhitungan

Tampilan hasil perhitungan berguna untuk mengetahui hasil dari perhitungan dokumen, query dan cosine similarity.

A. Hasil perhitungan Dokumen

User dapat mengetahui hasil perhitungan setiap kata dari tiap dokumen.



Hasil kandidat kata

No	Id skripsi	Term	Tf	Df	IDF	Bobot	Bobot kuadrat
1	1	Data	4	4	1.48	4.48	8.598
2	2	Aplikasi	3	2	1.47	4.08	8.118

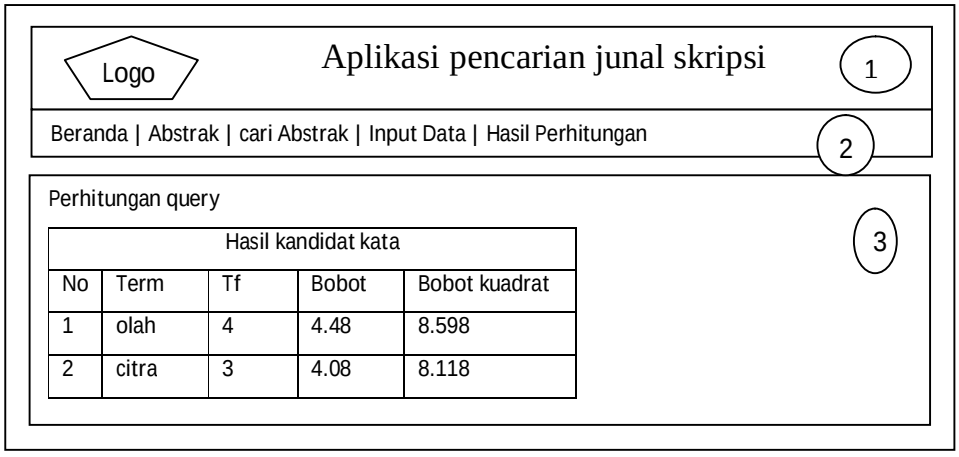
Gambar 3.22 Antarmuka hasil perhitungan dokumen

Berdasarkan gambar 3.22 :

1. Logo civitas akademika dan judul aplikasi
2. Label menu
3. Label daftar perhitungan dokumen

B. Hasil Perhitungan Query

User dapat mengetahui hasil perhitungan setiap kata dari inputan *query*.



Hasil kandidat kata

No	Term	Tf	Bobot	Bobot kuadrat
1	olah	4	4.48	8.598
2	citra	3	4.08	8.118

Gambar 3.23 Antarmuka hasil perhitungan query

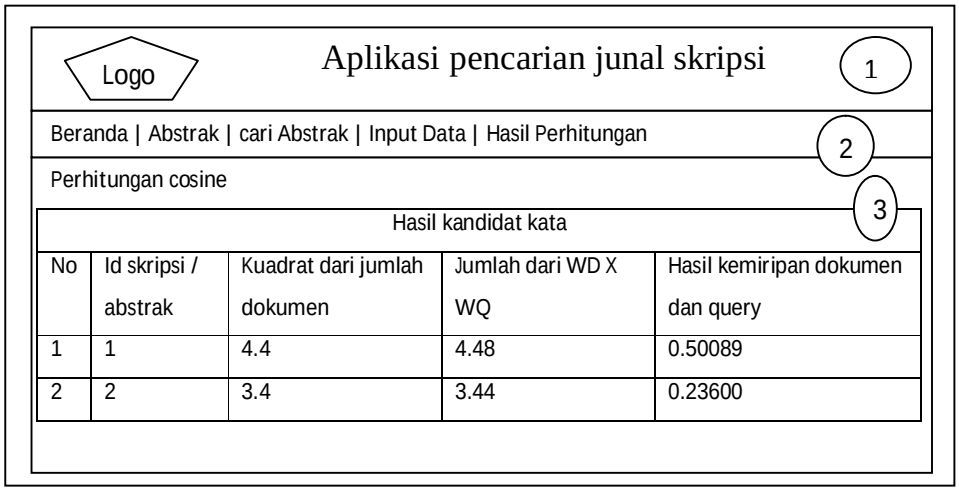
Berdasarkan gambar 3.23 :

1. Logo civitas akademika dan judul aplikasi
2. Label menu

3. Label daftar perhitungan query

C. Hasil Perhitungan Cosine

User dapat mengetahui hasil perhitungan setiap kata dari tiap dokumen.



Gambar 3.24 Antarmuka hasil perhitungan cosine

Berdasarkan gambar 3.24 :

1. Logo civitas akademika dan judul aplikasi
2. Label menu
3. Label daftar perhitungan dokumen